FR-THEP-07-01
SI-HEP-2007-07
arXiv:0708.4233

WHIZARD:[1]

Simulating Multi-Particle Processes at LHC and ILC

W. Kilian[a], T. Ohl[b], J. Reuter[c]

[a] *Fachbereich Physik, University of Siegen, D–57068 Siegen, Germany*
[b] *Institut für Theoretische Physik und Astrophysik, University of Würzburg, D–97074 Würzburg, Germany*
[c] *Physikalisches Institut, University of Freiburg, D–79104 Freiburg, Germany*

ABSTRACT

We describe the universal Monte-Carlo event generator `WHIZARD`. The program automatically computes complete tree-level matrix elements, integrates them over phase space, evaluates distributions of observables, and generates unweighted event samples that can be used directly in detector simulation. There is no principal limit on the process complexity; using current hardware, the program has successfully been applied to hard scattering processes with up to eight particles in the final state. Matrix elements are computed as helicity amplitudes, so spin and color correlations are retained. The Standard Model, the MSSM, and many alternative models such as Little Higgs, anomalous couplings, or effects of extra dimensions or noncommutative SM extensions have been implemented. Using standard interfaces to PDF, beamstrahlung, parton shower and hadronization programs, `WHIZARD` generates complete physical events and covers physics at hadron, lepton, and photon colliders.

[1] `http://whizard.event-generator.org`

Contents

1 The Need for Multi-Particle Event Generators

At the upcoming LHC and the future ILC experiments, we hope to uncover the mechanism of electroweak symmetry breaking and to find signals of physics beyond the Standard Model (SM). Many of the key elementary processes that have to be investigated for this purpose – such as weak-boson fusion and scattering, $t\bar{t}H$ production, supersymmetric cascades, exotica – are much more complex than the SM processes that were accessible at previous colliders. Simultaneously, the requirements for theoretical predictions of ILC processes will significantly surpass the LEP precision, while LHC data can only be meaningfully analyzed if a plethora of SM and possibly non-SM background and radiation effects are theoretically under control.

Traditional Monte-Carlo tools such as PYTHIA [1] or HERWIG [2] are able to predict signal rates for SM as well as various new-physics processes. These programs contain hard-coded libraries of leading-order on-shell matrix elements for simple elementary scattering, decay, and radiation processes. However, the requirements of precision and background reduction will only be satisfied if Monte-Carlo simulation programs can correctly handle off-shell amplitudes, multi-particle elementary processes, dominant radiative corrections, and matrix-element/parton-shower matching. Furthermore, the variety and complexity of proposed new-physics models makes it impractical to code every single process in a common library. There is obvious need for automated and flexible tools for multi-particle event generation, capable of simulating all kinds of physics in and beyond the SM.

This field has been pioneered by the CompHEP [3] and GRACE [4] collaborations, for processes of still limited complexity. The MadGraph [5] amplitude generator for the HELAS [6] library provided the first automatic tool for computing multi-particle amplitudes. In the last decade, the rapid increase in computing power together with the development of multi-channel integration techniques [7,8,9] has made multi-particle phase space accessible to Monte-Carlo simulation.

Furthermore, new ideas [10,11] have opened the path for a consistent inclusion of higher-order QCD effects in the simulation.

Consequently, several new approaches to the problem of realistic and universal physics simulation at the LHC and ILC have been implemented as event generators [12,13,14,9,15]. In this paper, we describe the current status of the WHIZARD [13] event generator package, which contains the O'Mega [16] matrix element generator and the VAMP [8] adaptive multi-channel multi-dimensional integration library.

2 Physics Simulation with WHIZARD

2.1 Purpose and Scope

WHIZARD is a program to compute cross sections and distributions of observables, and to generate simulated event samples, for scattering and decay processes of particles at high-energy colliders. The theoretical framework is set by leading-order perturbation theory, i.e., tree-level matrix elements. These are calculated in a fully automatic way. Refinements such as higher orders in perturbation theory, form factors, or other non-perturbative effects can be added manually.

The physics described by WHIZARD is given by the Standard Model of strong and electroweak interactions, and by well-known extensions of it, such as the minimal supersymmetric Standard Model (MSSM), Little Higgs models, anomalous couplings, and more.

The program covers physics at all experiments in elementary particle physics, including, for instance, LEP, Tevatron, the LHC, and the ILC. LHC physics is described by the convolution of parton distribution functions with hard-scattering processes. QCD effects that cannot be described by fixed-order perturbation theory are accounted for via standard interfaces to external programs. WHIZARD is particularly adapted to ILC physics due to the detailed description of beam properties (energy spread, crossing angle, beamstrahlung, ISR, polarization).

In contrast to classic event generators such as PYTHIA [1] or HERWIG [2], WHIZARD does not contain a fixed library of physics processes, and it is not limited to a small number of particles at the hard scattering level. Instead, for any process that is possible in the selected physics model, the matrix element is computed as needed and translated into computer code, using the O'Mega program. The O'Mega algorithm is designed to compute helicity amplitudes in the most efficient way, by eliminating all redundancies in the calculation as they would show up in a naive Feynman-graph expansion.

The other requirements for a complete event generator, in particular the phase space setup, are also implemented, and are also handled in a fully automatic way. The WHIZARD approach is thus similar to packages such as CompHEP [3], MadEvent [5,9], and Sherpa [15], which also aim at the simulation of multi-particle processes, but use different algorithms and implementations, and have different ranges of applicability and degrees of optimization.

2.2 Workflow

Using WHIZARD is quite easy. After the installation of the program which is described in sec. 6.1, a simple structure allows for a convenient configuration of the program by the user. The first

choice selects one of the physics models that are supported by WHIZARD, an overview of which can be found in sec. 6.2. After a specific physics model has been selected, the user can specify a list of processes for which the matrix elements are generated after a make command is executed on the command line. Note that the initial state specified by the user results in the selection of a partonic initial and final state at first. The user can decorate the partons by selecting structure functions for the particles in the initial state and adding showering, hadronization or dedicated decay packages for particles in the final state. After the process list has been completed, editing the initialization files allows for adjusting WHIZARD for Monte Carlo integration, event generation and analysis.

The user can then choose from the list of processes whether a single process or a set of processes are to be integrated in a single run of the program. Possible variables and switches that can be set for the integration contain the center of mass energy, whether the beams are structured or polarized (including specification of the corresponding structure functions), while WHIZARD automatically recognizes whether scattering processes or decays are considered. Central parameters of the physics model can be either set or read in from configuration files. WHIZARD supports the SUSY Les Houches Accord (SLHA) [17] which enables standardized input from SUSY spectrum generators.

In general, WHIZARD itself detects the complexity of the processes under consideration and estimates the number of necessary random number calls that are needed for a stable integration. This includes standard cuts that guarantee the absence of naive soft and collinear singularities. The way how the user can manipulate these settings will be described later in this paper. Furthermore, users can specify their own cuts, either for integration purposes (i. e. the cross section calculation takes care of the cuts, as well) or for the histograms.

For analysis purposes, event generation can be switched on. There are two options: weighted and unweighted events. The effort for unweighting the Monte Carlo events grows with the number of external particles, but is well under control. Weighted distributions need much less generation time, and result in smoother distributions, but their fluctuations do not correspond to real data with the same number of events. In addition, using weighted distributions in detector simulations can exercise the detector in regions of phase space that are thinly populated by real data, while scarcely probing regions of phase space where most of the real events will lie. The data output is available in several different event formats, ranging from a very long and comprehensive debugging format to machine-optimized binary format. Generated events are mandatory for analysis, i. e. for producing histograms. Histograms can easily be generated with WHIZARD's own graphics package, or standardized data files can be written out.

The WHIZARD executable can be called directly from the command line, as well as by executing a make command. It can be interfaced to external programs, which allows scanning or using WHIZARD as a driver for a (full) detector simulation.

2.3 Program Structure

The overall architecture of WHIZARD is sketched in figure 1. The structure of largely independent software components is both good programming practice and reflects the development history. WHIZARD [13], O'Mega [16] and VAMP [8] were developed independently and communi-

cate only via well defined interfaces. While O'Mega and VAMP were designed to solve only one problem, optimized matrix element generation (see section 4 for details) and adaptive multi channel importance sampling [8] respectively, the WHIZARD component plays a dual rôle, both as phase space generator and as the central broker for the communication among all the other components.

This component structure makes it possible to implement each component in a programming language most suited to its purpose:[1]

- VAMP is a purely numerical library and has therefore been implemented in fortran, making heavy use of the array features that were added to fortran with the fortran95 revision of the standard (an implementation in FORTRAN77 would have been impractical, in addition to the obsolescence of the standard).

- O'Mega has no numerical objectives, but is very similar to a modern retargetable optimizing compiler instead: it takes a model description, a description of a target programming language and set of external particles and generates a sequence of instructions that compute the corresponding scattering amplitude as a function of external momenta, spins, and other quantum numbers. For this purpose, (impure) functional programming languages with a strong type system provide one of the most convenient environments and O'Caml [18] was selected for this task. As a target programming language, only fortran95 is currently fully supported, since this is what is used in WHIZARD. Implementing descriptions of other target programming languages is straightforward, however.

- WHIZARD has again a strong numerical component and must be linked to VAMP. Therefore, fortran has been chosen for the implementation of the phase space generator.

These components interact via textual interfaces implemented in PERL scripts (it is planned to replace these PERL scripts in future releases by O'Caml code that is easier to maintain.) A reimplementation of all components in C++ would in principle be possible, but would represent a major undertaking due to the size of the program.

2.4 History

Work on WHIZARD began in 1998; its original purpose was the computation of cross sections and distributions for electroweak processes at a future linear collider. In particular, $2 \to 6$ fermion processes such as vector-boson scattering could not be treated by the automatic tools available at that time. The acronym WHIZARD reflects this: W, HIggs, Z, And Respective Decays. Since then, the scope of WHIZARD has been extended to cover QCD and hadron collider physics, the complete SM, and many of its extensions.

Initially, WHIZARD used MadGraph [5] and CompHEP [3] as (exchangeable) matrix-element generators. Subsequently, these have been replaced as default by the O'Mega optimizing matrix element generator which avoids the factorial growth of matrix-element complexity with the

[1] The choices made reflect the personal opinions of the authors on a subject that is often the realm of highly emotional arguments.

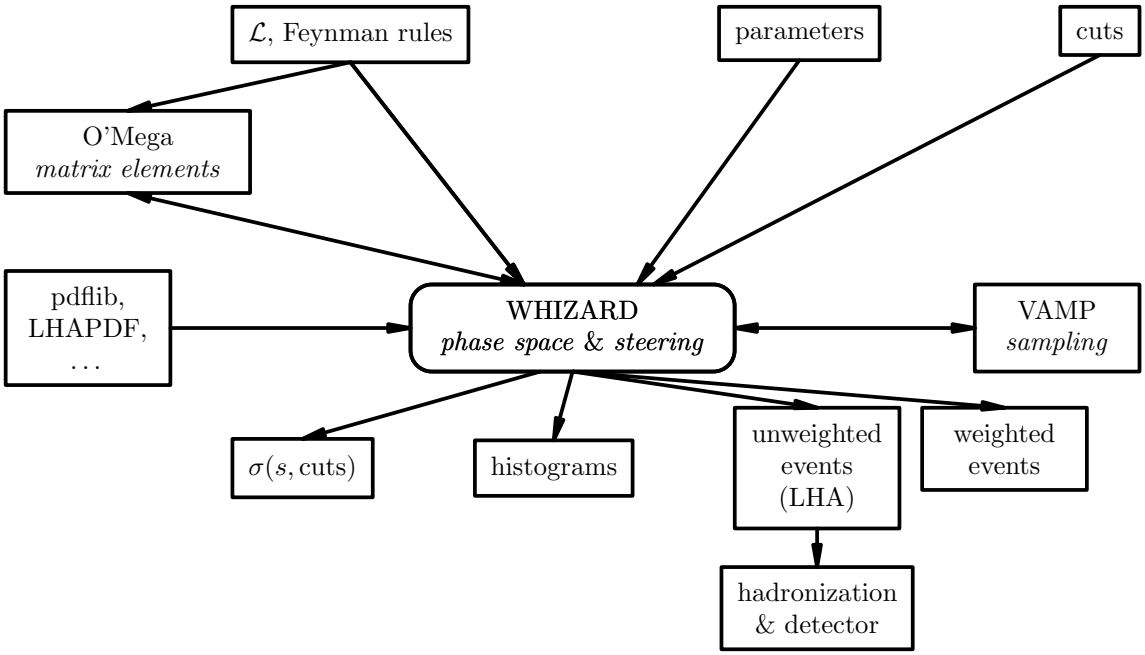

Figure 1: *The overall structure of* WHIZARD.

number of external particles. Furthermore, WHIZARD includes the VAMP [8] library for adaptive multi-channel integration. In its current state, the WHIZARD project has been merged with the VAMP and O'Mega projects.

3 Checks and Applications

3.1 Standard Model

WHIZARD supports the complete Standard Model of electroweak and strong interactions, and reliably computes partonic cross sections for processes with 4, 6, or more fermions in the final state, as they are typical for electroweak high-energy processes such as weak-boson, Higgs, and top-quark production. The correctness of the numerical results, while assured in principle by the validity of the underlying algorithm, nevertheless should be and has been checked, both by internal tests and in the context of published physics studies.

For instance, in recent work on W pair production [19], WHIZARD was used for numerically computing complete tree-level four-fermion cross sections, in agreement with analytic calculations. An exhaustive list of e^+e^- six-fermion cross sections in the SM has been carefully cross-checked between WHIZARD and LUSIFER [20]. All calculated cross sections were found to agree, taking into account differences in the treatment of unstable particles that are well understood. Six- and eight-fermion final states in top-quark processes have been studied in Ref. [21].

The determination of the Higgs potential will be one of the tasks for the next generation of

colliders. In a comprehensive study of Higgs pair production at the LHC and the ILC [22], the analytic results for SM Higgs pair production were numerically cross-checked with WHIZARD, and it could be established that triple Higgs production in the SM will remain below observability at either collider.

At SLAC, a large database of SM multi-fermion events in STDHEP format has been generated using WHIZARD [23], intended for further experimental and phenomenological ILC studies. A recent analysis of possible supersymmetry measurements at the ILC [24] made use of this database to demonstrate that SM backgrounds, calculated with complete tree-level matrix elements, are indeed significantly larger than predicted with the approximations of, e.g., PYTHIA.

Another application of WHIZARD in the context of the SM Higgs sector is the determination of the maximal significance with which several Higgs channels can be observed over background. This is achieved by a joint integration of the log-likelihood for signal and background over phase space [25].

3.2 Supersymmetry

Despite its conceptual beauty and simplicity, the minimal supersymmetric extension of the SM, the MSSM is a very complicated model with a large particle content and several thousand different vertices. Especially a vast number of mixing matrices and possible complex phases complicates any simple relations demanded by the symmetries of the MSSM. A comprehensive collection of all the Feynman rules of the MSSM and their derivation can be found in [26] and have been implemented in WHIZARD and O'Mega. The Feynman rules containing Majorana vertices are implemented according to [27].

In [28], comprehensive tests have been performed to verify that the implementation of the MSSM is correct. This has been done with the help of gauge and supersymmetry invariance checks (Ward and Slavnov-Taylor identities as described in [29,30]). To test all couplings that can be of any relevance for future experiments in particle physics phenomenology, a check of more than 500 different processes is necessary. This extensive list of processes has been tested in [28] by direct comparison with two other public multi-particle event generators, Sherpa [15] and MadGraph [5,9], showing accordance of all three programs within the statistical Monte Carlo errors. This extensive comparison which serves as a standard reference for testing supersymmetric processes can be found at http://whizard.event-generator.org.

With WHIZARD it was for the first time possible to perform simulations for SUSY processes with six final state particles using full matrix elements. For the ILC, the importance of off-shell effects was shown for the production of e.g. sbottom pairs in [28]. When using cuts to enhance the signal on top of the background, interferences of different production amplitudes (especially heavy Higgses and neutralinos) with identical exclusive final states lead to deviations from the narrow-width approximation by an order of magnitude. Similarly, sbottom production at the LHC with subsequent decay to a b jet and the LSP has been studied in [28]. There, also the contamination of the tagging decay jets by initial state radiation has been examined, amounting to the quite complicated process $pp \to \tilde{\chi}_1^0\tilde{\chi}_1^0 b\bar{b}b\bar{b}$ with more than 30,000 Feynman diagrams, several thousand phase space channels and 22 color flows. It was shown there, that off-shell effects are important for LHC as well, and secondly, that there is a severe combinatorial

background from ISR jets.

Projects that are currently in progress deal with the radiative production of neutralinos at the ILC [31], the measurement of SUSY CP phases at the LHC [32], the determination of chargino and neutralino properties at the ILC [33] as well as a complete cartography of the edge structures and spin measurements within the decay chains at the LHC [34]. Further projects deal with the implementation of extended GUT- or string-inspired supersymmetric models within WHIZARD to study their phenomenology [35].

3.3 Little Higgs

WHIZARD was the first multi-particle event generator in which Little Higgs models have been implemented. The Littlest Higgs and Simplest Little Higgs models are contained in the model library, including variants of these models like e.g. different embeddings of the fermionic content. Several studies for LHC and ILC as well as the photon collider have been performed with WHIZARD [36], focusing especially on the lightest available states in these models, the so-called pseudoaxions, η, pseudoscalar states which are electroweak singlets. The studies are concerned with production in gluon fusion at the LHC and detection via rare diphoton decays analogous to the light Higgs, to $t\bar{t}\eta$ associated production at the ILC, and investigations important for the model discrimination at the LHC and ILC using special types of couplings. Ongoing projects [37] deal with general search strategies at LHC, with focus on the heavy gauge bosons and the heavy fermionic partners of the quarks in these models. A brief overview of applications of WHIZARD in the context of Little Higgs models can also be found in [38].

3.4 Strongly Interacting Weak Bosons

If no new particles exist in the energy range of LHC and ILC, and even the Higgs boson is absent, some scattering amplitudes of electroweak vector bosons rise with energy and saturate unitarity bounds in the TeV range. This behavior should be detectable, both by the effects of new strong interactions, possibly resonances, at the LHC, and by anomalous couplings at lower energies at both colliders.

Improving on earlier studies [39] where due to computational restrictions, final-state W and Z bosons had to be taken on-shell, using WHIZARD it became possible to analyze distributions using complete tree-level matrix elements, vector-boson decays and all non-resonant background included. This allowed for detailed estimates of the ILC sensitivity to those couplings, taking into account all relevant observables including angular correlations, without the restrictions of on-shell approximations [40,41]. Also using WHIZARD for the simulation, an ongoing ATLAS study will clarify this issue for the LHC [42].

3.5 Exotica

WHIZARD has also been used to study top quark physics in higgsless models of electroweak symmetry breaking [43].

Even the phenomenology of very exotic models can be studied with WHIZARD, e. g. noncommutative extensions of the SM [44]. Noncommutative field theories can either be formulated as nonlocal field theories involving non-polynomial vertices or as effective field theories with polynomial vertex factors. In the first case, it is straightforward to add the corresponding vertex functions to omegalib, while in the latter case, care must be taken to consistently count the order of effective vertices.

For a study of exotic models that do not draw enough public attention to merit a complete supported implementation in O'Mega, the easily readable output of O'Mega, allows an alternative approach. After generating the SM amplitude with O'Mega in order to obtain a template with all the required interfaces in place, the corresponding fortran module can be edited with a text editor, replacing the SM vertices by the corresponding vertices in the model and adding diagrams, if required. The necessary fortran functions implementing exotic vertex factors can be added to the module, without having to modify omegalib, as discussed in sec. 7.2 below. More details about how to add models in general can be found in sec. 7.1.

4 O'Mega: Optimized Matrix Element Generator

O'Mega is the component of WHIZARD that constructs an optimized algebraic expression for a given scattering amplitude (or a set of scattering amplitudes) in a given model. While it can also be used by itself as a command line tool (e. g. for producing programs that plot differential cross sections), it is called by WHIZARD automatically with the correct arguments when a new process is added to an event generator.

4.1 Requirements

For complicated processes, such as searches for new physics at the LHC and a future ILC, the efficient perturbative computation of scattering matrix elements has to rely on numerical methods for helicity amplitudes, at least partially.

The time-honored trace techniques found in textbooks can reasonably only be used up to $2 \to 4$ processes and becomes inefficient for more particles in the final state. Therefore, in addition to allowing for polarized initial and final states, the computation of helicity amplitudes is the method of choice for complex processes. While there are very efficient methods for the analytical calculation of helicity amplitudes for massless particles, their extension to massive particles can become cumbersome, while, in contrast, a direct numerical evaluation results in the most efficient expressions to be used in cross section calculations and event generation.

It must be stressed that efficiency in the numerical evaluation of scattering amplitudes is not just a matter of potential wasteful use of computing resources. Instead, since the number of required CPU cycles varies over many orders of magnitude (cf. fig. 1), it can be the deciding factor whether simulations are possible at all.

In addition, all realistic quantum field theoretical models of particle physics employ the gauge principle at some point and are therefore subject to large numerical cancellations among terms in perturbative expressions for scattering amplitudes. Any implementation that fails

to group terms efficiently will suffer from large numerical uncertainties due to numerically incomplete gauge cancellations.

`O'Mega` implements an algorithm that collects all common subexpressions in the sum over Feynman diagrams contributing to a given scattering amplitude at tree level. Note that the common subexpression elimination (CSE) algorithm in a general purpose compiler will not be able to find all common subexpressions already in a moderately sized scattering amplitude, due to the size of the corresponding numerical expressions. It remains an open question, whether amplitudes calculated with twistor-space methods [45] could improve on the `O'Mega` algorithm (for a comparison that seems quite discouraging for twistor amplitudes, cf. [46]). While the former produce compact analytical expressions, substantial numerical complexity is hidden in the effective vertices. Furthermore, the extension to massive particles is not straightforward [47].

The building blocks used in `O'Mega` amplitudes correspond to expectation values of fields in states of on-shell external particles

$$W(x; p_1, p_2, \ldots, p_m) = \langle p_1, p_2, \ldots, p_n | \phi(x) | p_{n+1}, \ldots, p_m \rangle \ . \tag{1}$$

In the case of gauge bosons, they satisfy Ward identities, that ensure that gauge cancellations take place inside these building blocks [16].

4.2 Complexity

The irreducible complexity of a given tree level scattering amplitude is bounded from below by the number of its poles in kinematical invariants. In the absence of conserved charges, this number can be estimated by the number of independent internal momenta that can be built from the external momenta. Taking into account momentum conservation, it is easy to see that there are

$$P(n) = \frac{2^n - 2}{2} - n = 2^{n-1} - n - 1 \tag{2}$$

independent internal momenta in a n-particle scattering amplitude. This number should be contrasted with the number of Feynman diagrams. For realistic models of particle physics, no closed expressions can be derived, but in a one-flavor ϕ^3-theory, there are exactly

$$F(n) = (2n-5)!! = (2n-5) \cdot (2n-7) \cdot \ldots \cdot 3 \cdot 1 \tag{3}$$

tree level Feynman diagrams contributing to a n-particle scattering amplitude.

Obviously, $F(n)$ grows much faster with n than $P(n)$ (cf. table 1) and already for a $2 \to 6$ process, we find that the number of poles is two orders of magnitude smaller than the number of Feynman diagrams.

For realistic models with higher order vertices and selection rules at the vertices, empirical evidence suggests

$$P^*(n) \propto 10^{n/2} \tag{4}$$

while the factorial growth of the number of Feynman diagrams remains unchecked, of course.

While $P(n)$ is a priori a lower bound on the complexity, it turns out that this bound can approached in numerical [12,14] and symbolic [16] implementations. Indeed, the number of

n	$P(n)$	$F(n)$
4	3	3
5	10	15
6	25	105
7	56	945
8	119	10395
10	501	2027025
12	2035	654729075
14	8177	316234143225
16	32751	213458046676875

Table 1: *The number of ϕ^3 Feynman diagrams $F(n)$ and independent poles $P(n)$.*

independent momenta in an amplitude is a better measure for the complexity of the amplitude than the number of Feynman diagrams, since there can be substantial cancellations among the latter. Therefore it is possible to express the scattering amplitude much more compactly than by a sum over Feynman diagrams.

4.3 Ancestors

Some of the ideas that `O'Mega` is based on can be traced back to `HELAS` [6]. `HELAS` builds Feynman amplitudes by recursively forming off-shell 'wave functions' (1) from joining external lines with other external lines or off-shell 'wave functions' (1).

The program `MadGraph` [5] automatically generates Feynman diagrams and writes a Fortran program corresponding to their sum. The amplitudes are calculated by calls to `HELAS`. `MadGraph` uses one straightforward optimization: no statement is written more than once. Since each statement corresponds to a collection of trees, this optimization is very effective for up to four particles in the final state. However, since the amplitudes are given as a sum of Feynman diagrams, this optimization can, by design, never remove the factorial growth and is substantially weaker than the numerical algorithms of `ALPHA` [12] and `HELAC` [14] and the symbolic algorithm of `O'Mega` for more particles in the final state.

4.4 Architecture

`O'Mega` does not follow the purely numerical approach of [12,14], but constructs a symbolic representation of an optimally factored scattering amplitude instead, that is later translated to a programming language (`fortran95`) for compilation into machine code by a corresponding compiler. The symbolic approach brings three advantages:

1. the compilation to machine code allows a faster execution speed than the numerical programs that have to loop over large arrays. In practice this allows a two- to four-fold increase in execution speed, depending on the process and the model under consideration.

2. the intermediate `fortran95` code is human-readable and can easily be edited in order to experiment with the implementation of very exotic models, radiative corrections or form factors.

3. more than one programming language for the numerical evaluation can be supported.

For our applications, the second advantage ist particulary important.

Since it is an exclusively symbolic program, `O'Mega` has been implemented in the impure functional programming language `O'Caml` [18]. `O'Mega` makes extensive use of the advanced module system of `O'Caml` and is structured in small modules implementing abstract data types with well defined interfaces and strong type-checking.

The crucial feature of `O'Mega` is that it internally represents the scattering matrix element not as a tree of algebraic expressions, but as a Directed Acyclical Graph (DAG), where each subexpression appears only once, instead. In principle, it would be possible to construct first the tree corresponding to the sum of Feynman diagrams and to transform it subsequently to the equivalent minimal DAG by an algorithm known as Common Subexpression Elimination (CSE) in optimizing compilers. However, the size of the expressions resulting from the combinatorial growth with the number of external particles makes this all but impossible for practical purposes.

The basic algorithm of `O'Mega` proceeds as follows:

1. in a first step, functions from the `Topology` module construct the DAG corresponding to the sum over all Feynman diagrams in unflavored ϕ^n-theory, where n is the maximal degree of the vertices in the model under consideration. The abstract DAG data type is implemented by the `DAG` functor applied to a module describing the maximum degree of the vertices in the feynman rules of the model. It shoud be stressed that the algorithms in `O'Mega` place no limit on n and are therefore applicable to arbitrary effective theories and can support skeleton expansions of radiative corrections.

2. in a second step, all possible flavorings of the DAG are derived from the Feynman rules of the model encoded in a module implementing the `Model` signature. The algorithm ensures the symmetry and antisymmetry of the scattering amplitude for identical bosons and fermions. In the case of Majorana fermions, the standard algorithm for Feynman diagrams [27] has been implemented for DAGs as well [29]. Together with the numerical expression for each vertex and external state, this flavored DAG is the minimal faithful representation of the scattering amplitude.

   During this step, it is possible to select subamplitudes, e. g. by demanding that only contributions containing a certain set of poles are included. While this restriction to cascade decays can break gauge invariance because it doesn't select gauge invariant subsets [48], it is nevertheless a useful feature for testing the accuracy of commonly used approximations.

3. finally, a module implementing the `Target` signature is used to to convert the internal representation of the DAG (or of a set of DAGs) into code in a high level programming language (currently only `fortran95` is fully supported), that will be compiled and linked with the rest of `WHIZARD`.

This modular structure is supported by a library of purely functional abstract data types that implement, among others, combinatorial algorithms (`Combinatorics`), efficient operations on tensor products (`Products`) and the projective algebra of internal momenta subject to overall momentum conservation (`Momentum`).

The implementation of models as an abstract data type allows `O'Mega` to apply functors to a model and derive related models. For example, it is possible to automatically derive the R_ξ-gauge version of a spontaneously broken gauge theory from the Feynman rules in unitarity gauge [49]. Parsers for external model description files can also be implemented as a special case of `Model`.

5 Phase Space Integration and Event Generation

To promote a matrix-element generator such as `O'Mega` to a complete automated physics simulation program, it has to be supplemented by a variety of program elements. We have implemented all necessary parts in the program package `WHIZARD`. (This program name also stands for the event generator as a whole, including the matrix-element generating part.)

First of all, `WHIZARD` provides code that accounts for suitable phase-space parameterizations (phase-space channels), that selects among the multitude of possible channels, that integrates phase space with generic cuts in order to compute inclusive quantities, and that samples phase space in order to generate a sequence of exclusive (in particular, unweighted) scattering events. Since the computing cost of phase-space sampling, if done naively, can easily exceed the computing cost of optimized matrix-element computation by orders of magnitude, the algorithms must be able to keep this cost down at a manageable level.

Furthermore, `WHIZARD` implements (or contains interfaces to) the physics models, to beam dynamics such as parton structure, initial-state radiation, beamstrahlung, beam polarization etc., to final-state showering and hadronization, and to external event-file processing, detector simulation, and analysis.

The top-level routines that combine these parts to a working event generator, depending on user-provided configuration and input files, are also part of `WHIZARD`. Finally, `WHIZARD` contains a lightweight analysis module that can plot distributions of various physical observables without the need for external programs.

5.1 The `WHIZARD` Architecture

5.1.1 Overview

In the current implementation, `WHIZARD` is not a monolithic program but a program package that consists of several parts, written in suitable programming languages. Some of these parts are optional, some are exchangeable, some constitute integral parts of `WHIZARD`. When `WHIZARD` is installed, it includes all necessary and optional parts and provides a default setup that is ready for use, but can be adapted to more specialized needs if required.

At the top level, once the user has set up a configuration in terms of a physics model and a list of scattering and decay processes, this configuration is used for making and compiling a stand-

alone program, the actual event generator. Running this program calculates the corresponding observables and event samples. Like the complete package, the stand-alone generator is also called WHIZARD; actually it is an instance of WHIZARD that only implements the particular, user-selected list of processes for a particular physics model.

The top-level steering is accomplished by a UNIX-standard make sequence, i.e., the various steps are coded in terms of Makefiles which call the matrix-element generator, pass information to include files, compile process-specific and generic code, and link everything as the final program. This program can then be called directly, or alternatively called by further make commands that run the program and perform subsequent tasks such as plotting and displaying event distributions.

Some of the more complicated scripting tasks such as preparing the configuration and calling matrix element generators, moving and collecting files, and passing model-specific information to the WHIZARD user-interface routines, are implemented in PERL scripts that are automatically executed by the make commands.

5.1.2 Directory Structure

The main directory of the WHIZARD package in its current implementation[2] contains subdirectories where external programs and modules are stored and built upon request. These include the matrix element generators O'Mega, MadGraph (with HELAS), and CompHEP, the beamstrahlung code CIRCE, and the plotting tools FeynMF (Feynman diagrams) and gamelan (plots and histograms).

Some optional tasks are accomplished by standard external programs and libraries which are not included, but can be linked if they are accessible on the host. This applies to, e.g., PDF libraries, STDHEP, or PYTHIA.

The main directory also contains subdirectories where the WHIZARD source code is located, broken down in several parts (see below). A standard UNIX directory structure bin, lib, include serves the usual purpose of storing executables, libraries, and include files.

The compiled event generator, accompanied by templates for user-input files, is stored in a subdirectory named results. The set of files in this directory is complete: after compilation, it provides everything needed for running WHIZARD, so the WHIZARD instance can be duplicated, moved, and executed anywhere else.

5.1.3 The WHIZARD Core

The core WHIZARD code is written in fortran95. It consists of several parts, located in subdirectories, each of them broken down in modules. They are compiled and collected in three separate libraries, to be linked only in the final step: (i) the process-specific matrix element code; (ii) the VAMP integration package; (iii) the WHIZARD phase-space sampling code together with all auxiliary, interfacing, and user-level routines.

[2] It is foreseen that a future WHIZARD version will adopt a different structure, which transparently separates a static installation, possibly at a central location, from user- and project-specific code and data.

Since WHIZARD is written in a modular and, to a large extent, object-oriented style that makes heavy use of the type system, memory management, pointers, recursion, and array features of fortran95, there is no support for legacy FORTRAN77 compilers. It is foreseen that WHIZARD will gradually adopt the truly object-oriented features provided by the fortran2003 standard, once they have been implemented by the commonly used compilers. In particular, this will guarantee interoperability with C, so system-independent inclusion of C/C++ modules written by other parties in WHIZARD will be straightforward, and vice-versa[3].

Instead of using WHIZARD stand-alone, the libraries can also be linked to another program. This allows for interfaces to larger software frameworks, and to other programming languages.

5.2 Matrix Elements: O'Mega and others

The first step in setting up a WHIZARD event generator instance is the selection of a physics model and, for this model, a list of physics processes. When compilation is started, for each process in the list, an external matrix-element generator is automatically called which produces fortran code for the helicity amplitude. All references to these external routines are collected in a special module and compiled together with the code, to make up the process library that is finally linked with the WHIZARD executable.

The recommended matrix element generator is O'Mega, as described in the previous section of this paper. Since this generator produces optimized code for arbitrary processes in a large set of physical models, it suffices for most purposes. However, WHIZARD alternatively gives the opportunity to generate code using either MadGraph or CompHEP as matrix element generators, communicating via a well-defined interface. The requirements of this interface did not allow for supporting off-the-shelf versions of those programs, so the WHIZARD package contains specially adapted (but largely outdated) versions. Nevertheless, they are useful for special purposes, e.g., comparisons with the O'Mega result, since the algorithms are completely different and independent of each other and of the O'Mega algorithm. In particular, since CompHEP does not use helicity amplitudes but trace techniques, for cross sections of very simple processes ($2 \to 2$ and $2 \to 3$) the CompHEP-generated code turns out to be fastest if only the sum over helicities is required.

The interface supports introducing differently generated matrix element code, in particular hand-crafted code as it is needed for incorporating higher order radiative corrections or form factors. For those, no automated code generators suitable for exclusive event generation exist so far (but see Ref. [59] for a proof-of-principle implementation). Actually, there is a pseudo-matrix element generator "test" that merely provides unit matrix elements, useful for checking and plotting structure functions and phase space.

5.3 QCD and Color

While, in principle, the helicity state of a particle can be experimentally resolved, the color state of a parton is unobservable. Nevertheless, models for showering and hadronizing colored

[3]Since in terms of programming style, modern fortran supports a set of features similar to C++ and is more oriented towards computing-intensive applications, a recoding of WHIZARD itself in C++ will remain unlikely.

partons require that the program keeps track of color connections among final-state and, in the case of hadronic collisions, initial-state particles. This implies that much of the efficient color-algebra calculation methods commonly used for the analytical calculation of QCD amplitudes are not transferable to exclusive event generation, since the color connection information needed for parton-shower models is lost.

This fact is well known, and color connections are supported by classic Monte-Carlo generators such as `PYTHIA` and `HERWIG`. Since these programs, in most cases, construct amplitudes by combining $2 \to 2$ scattering processes with decay and showering cascades, keeping track of color is essentially straightforward. The problem becomes much more involved if many partons take part in the initial scattering, as it is the case for typical `WHIZARD` applications.

The original implementation of color in `MadGraph` makes color connections explicit: The amplitude is distributed among the possible color diagrams, which can be either squared including interferences (for the matrix element squared), or dropping interferences (for the parton shower model). This implementation, which requires some nontrivial color algebra in computing squares and interferences, is supported by the `WHIZARD`/`MadGraph` interface.

A purely numerical approach could also treat color as a random variable which is sampled for each event separately, or summed over explicitly. However, the individual colors of partons (e.g., "red", "green", "blue") are not directly related to the color connections used in parton shower and hadronization models, therefore this approach is not useful without modification. Sampling color connections as a whole would be possible; summing over them for each event leads back to the `MadGraph` algorithm.

The `WHIZARD`/`O'Mega` treatment of color takes a different road. It relies on the observation that the $SU(3)$ algebra of QCD is formally equivalent to a $U(3)$ algebra with a $U(1)$ factor subtracted [50]. In terms of Feynman graphs, in unitarity gauge, each $SU(3)$ gluon is replaced by a $U(3)$ gluon and a $U(1)$ ghost-gluon (gluon propagator with negative residue). The $U(1)$ gluon does not couple to three- and four-gluon vertices, so we have to include ghost diagrams just for gluon exchange between quarks. This idea leads to a very simple algorithm with the advantage that the color algebra becomes trivial, i.e., all color factors are powers of 3. Color connections are derived by simply dropping the ghost diagrams. Nevertheless, the squared matrix element computed by adding gluon and ghost diagrams exactly coincides with the matrix element computed in the $SU(3)$ theory.

5.4 Phase Space and Performance

For computing inclusive observables like a total cross section, the exclusive matrix element (squared) returned by, e.g., `O'Mega`, has to be integrated over all final-state momenta. If the initial beams have nontrivial structure, there are also integrations over the initial-parton distributions.

The integration dimension for typical problems (e.g., $2 \to 6$ scattering) exceeds 10, sometimes 20. Only Monte-Carlo techniques are able to reliably compute such high-dimensional integrals. This has the advantage that the algorithm can be used not just for integration, but also for simulation: a physical sequence of scattering or decay events also follows a probabilistic law. It has the disadvantage that the integration error scales with $c/\sqrt{N}$, where c is a constant,

and N the number of events. Since N is practically limited even for the fastest computers (a factor 10 improvement requires a factor 100 in CPU time), all efforts go into minimizing c.

For a uniform generation of physical events, in theory one should take a mapping of phase space that, via its Jacobian, transforms the kinematic dependence of the actual matrix element with its sharp peaks and edges into a constant defined on a hypercube. Such a mapping would also make integration trivial. Needless to say, it is known only for very special cases.

One such case is a scattering process defined by a single Feynman diagram of s-channel type, i.e., the initial partons fuse into a virtual particle which then subsequently decays. Here, phase space can be factorized along the matrix element structure in angles and invariant-mass variables, such that up to polynomial angular dependence, each integration depends on only one variable. For this case, we need mapping functions that transform a power law (massless propagator) or a Lorentzian (massive propagator) into a constant. Mapping polynomials is not that important; the angular dependence is usually not analyzed in detail and taken care of by rejection methods. The other two mappings provide the basis for constructing phase space channels in many algorithms, including the one of `WHIZARD`.

This simple construction fails in the case of t-channel graphs, where massless or massive particles are exchanged between the two initial particles. The overall energy constraint does not correspond to a line in the Feynman graph. However, the dependence of the exchanged propagator on the cosine of the scattering angle is still a simple function. Therefore, for finding a suitable parameterization, we flip the t-channel graph into a corresponding s-channel graph, where the z-axis of integration is aligned to one of the initial partons, so this polar angle becomes an integration variable.

Multiple exchange is treated by repeated application of this procedure. Flipping graphs is not unique, but for any choice it reduces t-channel graphs to s-channel graphs which typically also exist in the complete matrix element.

The main difficulty of phase-space sampling lies in the simultaneous presence of many, sometimes thousands, of such graphs in the complete matrix element. Neglecting interferences, one can attempt a multi-channel integration where each parameterization is associated with a weight which is iteratively adapted until it corresponds to the actual weight of a squared graph in the integral. This idea is adopted by the `MadEvent` algorithm. Since there are as many phase space channels as there are Feynman graphs, without further optimization the computing cost of phase space scales with the number of graphs, however.

Since the `O'Mega` algorithm results in a computing cost scaling better than the number of graphs, computing `WHIZARD` phase space should also scale better, if possible. To our knowledge, for phase-space integration there is no analog of the `O'Mega` algorithm that accounts for interference. Hence, `WHIZARD` uses heuristics to keep just the most important phase space channels and to drop those that would not improve the accuracy of integration. To this end, it constructs Feynman graphs for the process, keeping track of the number of resonances or massless branchings, and dropping terms that fail to meet certain criteria. The remaining number of phase space channels (which might come out between a few and several hundred) is then used as the basis for the `VAMP` algorithm which further improves the mappings (see below). After each `VAMP` iteration, the contributions of all channels are analyzed, and unimportant channels are dropped.

While this is not a completely deterministic procedure, with slight improvements and tunings it has turned out to be stable and to cover all practical applications. By construction, it performs well for "signal-like" processes where multiply-resonant Feynman graphs give the dominant contribution to the matrix element, and subdominant graphs are suppressed. In contrast to the PYTHIA approach which considers only the resonant graphs in the matrix element, WHIZARD does include all Feynman graphs in the matrix elements and returns the complete result. Only the method of integration takes advantage of the fact that dominant graphs dominate phase space.

"Background-like" processes like multiple QCD parton production without resonances, at first glance, appear to be not covered so well since the number of dominant graphs is not restricted and becomes large very quickly. This case has not been tested to the extreme with WHIZARD, although for $2 \to 6$ QCD processes it still gives stable results. However, fixed-order perturbation theory is not viable for a large number of partons (unless cuts are very strict, such that the cross section itself becomes unobservable), and parton-shower methods are suited better. With the caveat that proper matching of matrix element and parton shower is not yet implemented, we can conclude that the WHIZARD phase-space algorithm covers all cases where the fixed-order matrix element approximation is valid.

For a Monte-Carlo cross section result, the decisive performance criterion is the value of c in $\Delta\sigma/\sigma = c/\sqrt{N}$. After adaptation, in typical applications such as electroweak $2 \to 6$ processes, a WHIZARD run typically returns a number of order 1, so with 10^6 events a relative error in the per-mil range can be expected. (In simple cases the accuracy can become much better.)

Another important criterion for a Monte-Carlo algorithm is its ability to identify the maximum weight of all events, and the fraction of this maximum that an average event gives. This determines the reweighting efficiency for generating unweighted event samples, and if many events are required, the overall computing cost drastically depends on this efficiency.

WHIZARD keeps track of the reweighting efficiency. With WHIZARD's selection of phase space channels and VAMP's adaptive sampling, in applications with multiple partons and t-channel graphs it typically ends up in the per-mil- to percent range, while in favorable cases (multiply resonant, i.e., signal-like), efficiencies of order 10 % are common. Given the fact that for a meaningful cross section result, the number of events in the integration step is often a factor 100 higher than the number of unweighted events needed in the subsequent simulation, with efficiencies in this range the computing cost of adaptation, integration, and event generation averages out.

5.5 Multi-Channel Adaptive Sampling: VAMP

For multi-dimensional integration, WHIZARD makes use of the VAMP integration package [8]. The VAMP algorithm is an extension of the VEGAS algorithm [51]. The VEGAS algorithm introduces a multi-dimensional rectangular grid in the integration region. For each iteration, a given number of events (e.g., 10^6) is distributed among the cells, either on a completely random basis (importance sampling) or evenly distributed among the grid cells, but randomly within each cell (stratified sampling). For stratified sampling, usually the number of cells of the original grid (e.g., 20^{15}) is too large for filling each of them with events, so an auxiliary super-grid with

less cells is superimposed for this purpose (pseudo-stratification), and within each super-cell, the events randomly end up in the original cells.

After each integration pass, the sum of integrand values, properly normalized, yields an estimator for the integral, and the binning of each dimension is adapted to the shape of the integrand. For importance sampling, the adaptation criterion is the integral within each bin, while for stratified sampling, the bins are adapted to the variance within each bin. In practice, for high-dimensional Feynman integrals both importance sampling and stratified sampling give results of similar quality.

The `VAMP` algorithm [8] combines this method with the technique of multi-channel sampling [7]. All selected phase-space parameterizations, properly mapped to the unit hypercube, are sampled at once, each one with its own `VEGAS` grid. The estimator of the integral is given by the weighted sum of the individual estimators, where the weights α_i are initially equal (with $\sum \alpha_i = 1$), but are also adapted after each iteration, according to the channel-specific variance computed for this iteration.

The `VEGAS` algorithm has the effect that peaks in the integrand, during the adaptation process, become flattened out because after adaptation more events are sampled in the peak region than elsewhere. This works only for peaks that are aligned to the coordinate axes. Using `VAMP`, one tries to arrange the parameterizations (channels) such that each peak is aligned to axes in at least one channel. Since the integrand in any channel is corrected by the Jacobian of the transformation to the other channels (see Ref. [8] for details), in effect peaks are removed from all channels where they are not aligned, and flattened out in those channels where they are. As a result, after adaptation, within each channel the effective integrand is well-behaved, and both the integration error and the reweighting efficiency are significantly improved.

This adaptation proceeds on a statistical basis, and for reasonable numbers of events and iterations it is a priori not guaranteed that an optimum is reached. In particular, fluctuations become overwhelming when the number of channels, i.e., degrees of freedom, becomes too large. However, with the selection of phase-space parameterizations done by `WHIZARD`, the algorithm has proved sufficiently robust, such that it is universally applicable to the physics processes that `WHIZARD` has to cover.

5.6 Event Generation

With `WHIZARD`, simulated events can be generated after several adaptive iterations have resulted in reasonably stable `VAMP` integration grids, and a number of final iterations have yielded a reliable estimate for the total cross section. The `VAMP` grids are then fixed, and an arbitrary number of further events is generated. For each event, a first random number selects one of the possible channels, taking its relative weight into account, and within this channel a point is selected by importance sampling, taking the adapted binning into account. The event is kept or rejected according to the ratio of the integrand at this point compared with the channel-specific maximum weight. This results in a sequence of events that simulate an actual experiment.

Alternatively, for plotting distributions with greater accuracy, the weighted events can be recorded as-is.

Since the estimate for the maximum weight can only be determined by statistical sampling,

the reweighting – like any other statistical method – cannot exclude that the integrand value for a particular event exceeds this maximum estimate. This could be taken into account by again reweighting the whole sample according to the new maximum estimate. However, since WHIZARD is set up to put out unweighted events directly, we have chosen to merely record these excess events and to compute, at the end, the value of the error introduced by this excess. It turns out that in practice, this error is sufficiently below the overall integration error and can be ignored – if desired, it is possible to plot distributions of excess events and check for critical regions where the adaptation process could have failed.

5.7 Interfaces

So far, we have described WHIZARD as an event generator that is able, for fixed collider energy, to compute the partonic cross section for a scattering process, or a partial width for a decay process, and to generate simulated partonic events for this process. Actually, while the adaptation and integration proceeds separately for each process selected by the user, in the event generation step an arbitrary number of processes can be mixed.

For a complete physics simulation, this is not sufficient. First of all, in realistic colliders the partonic c.m. energy is not fixed. At hadron colliders, this energy is distributed according to parton distribution functions (PDFs). At lepton colliders, it is distributed according to the beam energy spectrum, affected mostly by beamstrahlung. Furthermore, initial-state radiation (ISR) reduces the available partonic energy. To account for this, WHIZARD is able to include the partonic energy spectrum in the integration. Each spectrum or radiation effect introduces an extra energy variable and thus increases the integration dimension by one. Since several effects may have to be convoluted (e.g., beamstrahlung + ISR), the number of extra integrations may be larger than two.

For computing these effects, WHIZARD makes use of external programs and libraries. While electromagnetic ISR is accounted for internally, for beamstrahlung and photon-collider spectra it interfaces CIRCE/CIRCE2 [52]. To account for generic e^+e^- energy spectra, WHIZARD can read events from GuineaPig [53] output. PDFs are taken from the standard PDFlib [54].[4]

Parton-shower, i.e., QCD radiation is not yet accounted for internally by WHIZARD. However, WHIZARD respects the Les Houches Accord [56] and therefore can interface to parton-shower Monte Carlo programs. For convenience, the PYTHIA Monte Carlo can be directly linked. This allows not just for showering partons (assuming that double-counting is excluded, i.e., the hard WHIZARD process does not include parton radiation), but also for interfacing hadronization, underlying events, and in fact adding all processes that can be simulated by PYTHIA as extra processes in the WHIZARD event generation sequence.

Alternatively, the generated event samples can be written to a file in various formats. These include human-readable as well as binary formats. For instance, WHIZARD supports the STDHEP format for event files.

[4] A LHAPDF [55] interface will be publicly available with the next WHIZARD version.

6 User Interface

6.1 Installation and Prerequisites

The WHIZARD package and the O'Mega matrix element generator are available as .tar.gz files[5] via

http://whizard.event-generator.org

This includes several auxiliary packages (VAMP integration, CIRCE beamstrahlung, etc.). Two compilers are needed: (i) a fortran95 compiler[6] for WHIZARD; (ii) the O'Caml compiler[7] [18] for O'Mega.

For hadron-collider applications, the PDFlib should be available on the system. The same holds for PYTHIA (hadronization etc.) and STDHEP, if these features are needed.

Installing and compiling WHIZARD proceeds via automatic configure and make commands. The physics configuration, i.e., the selection of the physics model and of the process list, is accomplished by editing a single file whizard.prc before compilation. The actual generator run is controlled by a further input file (plus a cut-definition file), to be edited by the user.

6.2 Physics Models

The support for specific models in WHIZARD relies on the implementation of the corresponding models in O'Mega and WHIZARD. In both packages, the infrastructure supports the incorporation of particles with spin 0, $\frac{1}{2}$ (Dirac/Majorana fermions), 1, $\frac{3}{2}$ and 2. Since the structure of O'Mega allows for the incorporation of arbitrary higher-dimensional operators all possible physics models based on quantum field theories containing particles with spins up to two can be implemented; even more general setups are possible like models based on noncommutative generalizations of space-time (see below).

Specific physics models are defined with the help of their particle content (and the corresponding quantum numbers), the fundamental interactions (vertices) and – most importantly and error-prone – the set of coupling constants and parameters together with the relations among them. Within O'Mega, some basic toy models like QED and QCD as well as the SM and its derivatives (like non-unitary gauges, extensions with anomalous couplings with and without K matrix unitarization, non-trivial CKM matrix) are implemented in models.ml, the supersymmetric models like the MSSM and possible extensions (NMSSM, ESSM etc.) in models2.ml, while non-SUSY BSM extensions (like Little Higgs models, Z' models and extra dimensional models) are implemented in models3.ml. In this module there is also a model Template which has exactly the same content as the SM, but can be augmented by the user

[5] The packages are designed for UNIX systems, LINUX in particular. Other operating systems may also be supported in the future.

[6] Some compilers are still buggy and fail to compile WHIZARD; consult the WHIZARD website or contact the authors for details.

[7] O'Caml is part of most standard LINUX distributions; otherwise it is available free of charge from http://caml.inria.fr/ocaml/.

Model type	with CKM matrix	trivial CKM
QED with e, μ, τ, γ	–	QED
QCD with d, u, s, c, b, t, g	–	QCD
Standard Model	SM_CKM	SM
SM with anomalous couplings	SM_ac_CKM	SM_ac
SM with K matrix	SM_km_CKM	SM_km
MSSM	MSSM_CKM	MSSM
Littlest Higgs	—	Littlest
Littlest Higgs with ungauged $U(1)$	—	Littlest_Eta
Simplest Little Higgs (anomaly-free)	—	Simplest
Simplest Little Higgs (universal)	—	Simplest_univ
SM with spin-2 graviton	—	Xdim
SM with gravitino and photino	—	GravTest
Augmentable SM template	—	Template

Table 2: *List of models that are currently supported by WHIZARD: the SM and its relatives, simple subsets of the SM, the MSSM, other models beyond the SM as well as a template which can be augmented by the user to include additional new particles and interactions.*

to incorporate new particles and interactions in his or her favorite model. More details about how this works can be found in subsection 7.1.

In WHIZARD for each model MODEL there is a file MODEL.mdl which contains the particles with their quantum numbers (electric charge, color etc.) as well as a definition of the basic parameters that can be accessed via the input file. This file also contains a list of all the vertices of the model, which is important for the generation of phase space of processes in that specific model. For each model MODEL, there is also a file parameters.MODEL.omega.f90 which contains all the couplings of the corresponding model as functions of the basic input parameters. An overview over the publicly supported models as well as those currently in their testing phase are shown in Table 2.

A next step will be to read in model definitions from programs that automatically calculate Feynman rules from more or less arbitrary Lagrangians, like LanHEP [57]. An experimental version of O'Mega is able to derive Feynman rules from Lagrangians by itself; later on, this will migrate into the official version. Already now, O'Mega can read in model files and generate its own matrix element generator generator for that specific model. This avoids the necessity of having programming skills in O'Caml, but also lacks the advantages of the functional approach of O'Caml, e.g. the availability of functors.

6.3 Processes

For a given physics model, WHIZARD can compute cross sections or partial decay widths for all processes that are physically allowed. The user-specified list of processes can be arbitrary,

as long as the computer is capable of dealing with it.[8] For each process, the O'Mega matrix-element generator (or, alternatively MadGraph/CompHEP) generates a tree-level matrix element, so without manual intervention, the result corresponds to fixed leading order in perturbation theory.

To define a process, the user may completely specify incoming and outgoing particles, choosing from the elementary particles contained in the selected model. For convenience, it is possible to define particle aliases and to sum over particles in the outgoing state, e.g., combine all neutrino generations or all light quarks.[9] In this case, all contributing matrix elements will be added at each selected phase-space point; for the generated events, a particle combination will be selected event by event according to the relative weight of the corresponding squared matrix element.

For the processes, a few options are available: for instance, the user can restrict intermediate states to select or exclude classes of Feynman graphs, and he or she can specify the treatment of unstable particles with regard to gauge invariance. Otherwise, setting parameters, cuts etc. is done on a per-run basis.

6.4 Beams and Partons

Once the event generator has been compiled, all processes in the user-specified list are available for calculating cross sections (with arbitrary cuts) and simulating events. The input file whizard.in contains the information that can be specified by the user.

The user selects a list of processes among the available ones and specifies the type and energy of the colliding beams (or the type of decaying particle). Each beam can be given a structure or polarization. For instance, in hadron collisions, the beam structure is given by the PDF set, specified by the usual PDFlib or LHAPDF parameters. Lepton collisions are affected by beamstrahlung and electromagnetic initial-state radiation. Photon collisions proceed via CIRCE spectra or via photons radiated from leptons in the effective-photon approximation. In all cases, all free parameters can be set and modified in the input file.

Apart from these physical beam setups, it is possible to compute fixed-energy cross sections, partial widths, and event samples for any types of colliding or decaying particles.

In lepton and photon collisions, polarization is of importance. For each beam, the user can specify the longitudinal polarization or, alternatively, transversal polarization. Since helicity amplitudes are used throughout the program (O'Mega), the polarization of the final-state particles can also be extracted.

[8] Hard-coded restrictions for processes with more than eight final-state particles exist but will be removed in an upcoming version; note however, that the CPU time needed for compilation and running considerably depends on the process complexity.

[9] This feature is currently restricted to the outgoing state and to particles of equal spin; the restrictions will be removed in an upcoming version.

6.5 Parameters, Cuts, and Other Input

WHIZARD follows the philosophy that no numerical parameters are hard-coded, everything can be specified by the user. With the caveat that some parameter relations are fixed by the model definition (to ensure gauge invariance), all free physics parameters such as particle masses, widths, and couplings can be modified in the input file. This also implies that the phase-space setup, which depends on particle masses, is generated afresh for each WHIZARD run.

For supersymmetric models, there is the SLHA standard [17] which specifies how to transfer physics parameters between programs. A SLHA file can either be included in the WHIZARD file, or read separately.

Cuts on phase space are of particular importance. Since many cross sections are infinite if no cuts are applied, WHIZARD by default generates a set of cuts on invariant masses and momentum transfer that is applied where it is necessary. The user can either reset the default cut values, or instead specify his/her own set of cuts[10], which then overrides the default ones. For specifying user cuts, a wide range of observables such as p_T, angles, rapidity, etc. is available.

The input file furthermore contains parameters that control the WHIZARD run, modify the phase space setup, affect diagnostics output, and more.

All parameters can also be set on the command line. This enables, for instance, scanning parameters by looping over WHIZARD runs in a shell script.

6.6 Using and Analyzing Results

The WHIZARD user interface has been designed with various applications in mind, ranging from theoretical studies to experimental simulation.

A theoretical study typically implies the calculation of some cross section and the display of characteristic distributions of observables. To this end, the user would set up the processes and parameters, run the program to compute cross section integrals, and generate a sufficiently large sample of weighted events. In this case, one would not use the rejection algorithm to unweight events, so no information is lost. It is possible to write the event sample to file and to do analyses by some external program, but WHIZARD also contains its own analysis module. With this module, the user specifies lists of observables to histogram (on top of, and analogous to specifying cuts). During event generation, the program will fill those histograms and output data tables. To plot such data, WHIZARD employs the gamelan package. This program generates encapsulated PostScript code that can conveniently be included in LaTeX documents.

For a simulation study, the user needs a sequence of unweighted, fully hadronized events. The WHIZARD run includes the necessary steps of adaptation and integration and proceeds to the generation of unweighted events; the event sample may be specified either by the number of events or by an integrated luminosity. Hadronization is accomplished by linking PYTHIA or some other hadronization package to WHIZARD, either internally, or externally by post-processing event files. WHIZARD supports several event file formats, including the STDHEP binary format. These event samples are ready to be further processed by detector simulation and analysis.

[10] This is done in a separate cut-definition file. In an upcoming version, this will be merged with the main input file.

It is often necessary to re-run a program several times in order to change or refine event numbers, analysis parameters, etc. Since adaptation, integration, and event generation all can take considerable time, WHIZARD provides means for reusing unfinished or previous integration results, grids, and events, so the program needs not start from the beginning. The integrity of data is checked by MD5 sums. Furthermore, for the use within batch systems with their restrictions, there are several handles for limiting running time, splitting output files, etc.

6.7 Calling From Other Programs

While WHIZARD is a stand-alone event generator, it is sometimes useful to have a library of subroutines that can perform the same tasks within a larger computing environment. This is, for instance, the approach taken by PYTHIA where no main program is contained in the core distribution.

This functionality is also available for WHIZARD. With the usual configuration set up, calling `make lib` instead of the default `make` generates just the subprogram libraries. These can be linked to any program that is capable of calling `fortran` subroutines. The library API accounts for initialization and finalization of the program, the adaptation/integration stage, and various options for event generation. Parameters can be set either via the standard input files, or via a designated subroutine. Apart from screen and file output, cross section results and error estimates can be retrieved directly.

As an example, a JAVA interface to WHIZARD has been presented in Ref. [58].

7 Extensions and Extensibility

7.1 Building Models

Adding a new model to WHIZARD is straightforward. As has been discussed in subsection 6.2, to add a new model in the hard-coded version of the programs, one has to edit both WHIZARD and the O'Mega driver simultaneously. In the file `models3.ml` in the `src` directory of O'Mega there is a `Template` module which is just a copy of the SM implementation within O'Mega. From this template one can read off the syntax structure and add new particles and interactions according to one's favorite new physics model. New coupling constants have then to be declared in the Fortran file `omega_parameters_template.f95` [11]. This exhausts the changes that have to be made on the O'Mega side.

The next step is to add all new particles with their quantum numbers in the file `Template.mdl` in the subdirectory `conf/models/` of WHIZARD. In the bottom part of that file all new interaction vertices have to be added in the way of the SM vertices already written down there. This is important in order that WHIZARD can find the phase space channels including the new particles added by the user. The hardest and most error-prone work to do is to add the functional

[11] If the `noweb` literate programming tool is available, it is beneficial to edit the file `omega_parameters_template.nw` instead, since it allows for more comprehensive documentation, in particular of mathematical relationships.

relations among the coupling constants and parameters beyond the SM within the corresponding parameter file `parameters.TEMPLATE.omega.f90` in the same directory, `conf/models` of `WHIZARD`. Again, the examples from the SM might serve as a guideline here for the way how to incorporate all the new couplings in this file. The model `Template` can be accessed in `WHIZARD` with the tag `Template` in the same way as the other models defined in subsection 6.2. It can even be used when the user has not added any new particles or interactions; in that case it is just a mirror of the SM.

7.2 Advanced use of `WHIZARD`

The source code generated by `O'Mega` is very easy to read and consequently also to modify. E. g., fig. 2 shows the complete $e^+e^- \to \mu^+\mu^-$ scattering amplitude in the SM. Notice that, for convenience, the crossed amplitude with all particles outgoing is calculated internally. For this reason the incoming momenta are reversed.

In the code, `mass` is an array of particle masses, indexed by the PDF Monte Carlo particle codes, that is defined in the module `omega_parameters`. `qlep`, `gnclep(1)` and `gnclep(2)` are the lepton charge and vector and axial vector neutral current coupling constants, respectively. `wd_tl` is a function that returns a non-zero width for time-like momenta. The functions `pr_feynman` and `pr_unitarity` implement the propagators and the functions `v_ff` and `va_ff` implement vector couplings and mixed vector/axial couplings of the fermions given as arguments.

It is now straightforward to replace any of these functions by another function that computes a non-standard propagator or coupling or to add another particle exchange, like a Z'. Of course, it is more efficient for a comprehensive analysis of a Z'-model to produce a new model file, but non-standard vertices are a useful hook for adding radiative corrections (see sec. 7.3). When preparing modified vertex factors for fermions, it is most convenient to use the elementary vertex factors for the γ-matrix structures, as they are already optimized and guaranteed to be consistent with the conventions used in the other functions from `omegalib`.

Note that the final line, `amp = - amp`, takes care of all the factors of i coming from vertices and propagators in the Feynman rules. Any modification of the amplitude must respect this convention, in order not to spoil potential interference terms.

Since recompilations in `WHIZARD` are controlled by `make`, it is advised not to edit the module file containing the scattering amplitude in place, but in a separate directory and to copy them to the `WHIZARD` directory afterwards. Otherwise, mismatched modification times, as are common in network file systems, might cause `make` to overwrite the edited file.

7.3 Higher Orders

To match the experimental precision of hadron and lepton collider environments theoretical predictions have to include higher order radiative corrections, originating from virtual and real diagrams. Hard real radiation (fixed order) is a very natural feature of a multi-particle event generator that is automatically built in `WHIZARD`. All NLO applications as yet have been performed in a way such that a pre-calculated fixed-order NLO matrix element has been imple-

```
! process: e+ e- -> mu+ mu-
pure function l1bl1l2bl2 (k, s) result (amp)
  real(kind=omega_prec), dimension(0:,:), intent(in) :: k
  integer, dimension(:), intent(in) :: s
  complex(kind=omega_prec) :: amp
  type(momentum) :: p1, p2, p3, p4
  type(spinor) :: l2_3, l1_2
  type(conjspinor) :: l2b_4, l1b_1
  type(vector) :: a_12, z_12
  type(momentum) :: p12
  p1 = - k(:,1) ! incoming e+
  p2 = - k(:,2) ! incoming e-
  p3 =   k(:,3) ! outgoing mu+
  p4 =   k(:,4) ! outgoing mu-
  l1b_1 = vbar (mass(11), - p1, s(1))
  l1_2 = u (mass(11), - p2, s(2))
  l2_3 = v (mass(13), p3, s(3))
  l2b_4 = ubar (mass(13), p4, s(4))
  p12 = p1 + p2
  a_12 = pr_feynman(p12, + v_ff(qlep,l1b_1,l1_2))
  z_12 = pr_unitarity(p12,mass(23),wd_tl(p12,width(23)), &
     + va_ff(gnclep(1),gnclep(2),l1b_1,l1_2))
  amp = 0
  amp = amp + a_12*( + v_ff(qlep,l2b_4,l2_3))
  amp = amp + z_12*( + va_ff(gnclep(1),gnclep(2),l2b_4,l2_3))
  amp = - amp ! 2 vertices, 1 propagators
  ! unit symmetry factor
  ! unit color factor
  ! Number of external adjoints: 0
end function l1bl1l2bl2
```

Figure 2: ***fortran95*** *code generated by* ***O'Mega*** *for the SM* $e^+e^- \to \mu^+\mu^-$ *scattering amplitude. In this example, the code is equivalent to the sum of Feynman diagrams, as there are no common subexpressions.*

mented in WHIZARD, either as an external subroutine or an external library. In [59] precompiled NLO matrix elements for the production of two SUSY particles (mainly charginos or neutralinos) at the ILC have been used linked to WHIZARD as a so-called user-defined structure function.

The most tricky part in including NLO matrix elements within any event generator is the correct matching between the fixed-order NLO matrix elements and the parton showers. The fixed-order part contains the virtual corrections together with the soft (including soft-collinear) parts of the real emission united in the so-called soft-photon factor. The parton shower, on the other hand, resums (in the case of a lepton collider) soft emission to all orders using the Gribov approximation [60] and hard collinear radiation to third order [61]. This task has been performed in [59] where the phase-space slicing method has been used to subtract regions of phase space in order to avoid double counting. The fixed-order NLO matrix elements in [59] have been calculated in [62] (and recalculated and confirmed in [59]) with the help of the FeynArts, FormCalc and LoopTools packages [63]. With the interface that has been used for the implementation of these NLO matrix elements in [59] it is possible to use general one-loop NLO results from FeynArts/FormCalc/LoopTools in WHIZARD. So in principle all two-sparticle production processes at the ILC are available as general Monte-Carlo event generators within the WHIZARD framework. For the technical details about the implementation and the interfacing we refer to a later publication [64]. As discussed in the next subsection, one of the next upgrades of WHIZARD will be able to force the phase space on-shell and allow subsequent decay chains of one-shell produced particles (although this might be inconsistent for beyond the SM models, as pointed out in subsection 3.2). Then it is a straightforward task to include the decays of the sparticles to simulate (at least in a double-pole approximation) the full SUSY process with subsequent decays at NLO with WHIZARD. This is work in progress [33].

Another ongoing project along these lines deals with the precision measurement of the Higgs self coupling which will be performed best at a future photon collider [65]. The leading order process is here already at the one-loop level. In order to have a gauge-invariant result the full one-loop triangle and box graphs have to be considered. Since an off-shell continuation of these matrix elements which are defined only on-shell will break gauge invariance this project relies again on a recursive on-shell phase space generation.

A remark concerning the parton showers discussed above: For QED the whole shower (to the abovementioned precision) is implemented as one analytic expression within WHIZARD and can be simply switched on or off. In the case of LHC, QCD showering has to be applied to colored particles which can be done with the PYTHIA implementation within WHIZARD. More about the implementation of QCD parton showers can be found in the next subsection.

7.4 Future Features

In its current state, the WHIZARD Monte-Carlo generator accounts for optimized complete tree-level matrix elements of partonic cross sections and for efficient phase-space integration and event generation in realistic applications. However, real events do not consist of partons, and multiple QCD radiation that cannot be treated at fixed order plays an important role.

WHIZARD does generate complete hadronic events. It does not internally shower or hadronize

partons, it delegates these tasks to external programs. This is not a problem since the standard Les Houches interface [56] is used for transferring particle and color-connection data. However, the present implementation forces the user to decide among two mutually exclusive possibilities: (i) compute the complete matrix element and phase space with a fixed strong coupling for as many partons as feasible and transfer this directly to hadronization, or (ii) do not radiate any partons within `WHIZARD` and activate the external parton shower, i.e., use the collinear approximation for all extra quarks and gluons.

Naively combining a multi-parton matrix element with the external parton shower can easily lead to double-counting. On the other hand, there are algorithms which systematically avoid this problem at leading order [10,66], and recently it has been shown how it can be solved to all orders in perturbation theory [67]. For a reliable quantitative treatment of QCD effects at the LHC in particular, the implementation of such an algorithm is mandatory, and will be implemented in a future version of `WHIZARD`.

A similar problem arises for MSSM and other new-physics processes: while it is desirable to compute decay cascades of supersymmetric particles off-shell with complete matrix elements, for complex cascades with 12 or more particles this is neither computationally feasible, nor necessary, given experimental uncertainties at the LHC. Therefore, possibilities to partially use on-shell approximations and treat particle decays on an inclusive level, while generating exclusive events, will be implemented and merged with the standard `WHIZARD` approach.

The high requirements on precision call for the inclusion of loop corrections, again suitably matched to parton radiation, in exclusive event generation. As discussed in Sec. 7.3, a first step in that direction has been done for `WHIZARD`. In the future, there will be an increased effort to include those effects, preferably on an automated basis.

While the overall speed and efficiency of `WHIZARD` has proven satisfactory for many purposes, there are possible applications where improvements are welcome. The matrix-element evaluation code produced by `O'Mega` is optimized as far as redundant arithmetic operations are concerned, so significant improvements are not expected. In typical applications, phase space is not a bottleneck either. However, significant optimizations can be expected by parallel evaluation of critical parts of the program. Monte-Carlo sampling algorithms can, in principle, be parallelized in a straightforward manner, and the `VAMP` module already has full support for parallel execution. It is planned to extend this support to all critical parts of the `WHIZARD` system.

Currently, `WHIZARD` follows the traditional style of programs that read their input from text files and are controlled on the command line. This approach ensures maximum transparency and flexibility. However, with increasing complexity of the program the alternative possibility of steering the program via a graphical user interface becomes desirable, and is planned for a future version.

In the current implementation of `O'Mega`, all models are described by tables of vertices in `O'Caml` syntax and by `fortran` functions that compute all coupling constants and particle masses from the parameters of the models. Neither syntax requires specialized knowledge for extending a model or implementing a new one from scratch. Nevertheless a new syntax that is closer to the usual notation used in physics papers will make phenomenological studies of models for new physics even more straightforward. In addition, further parsing modules will

allow reading of existing model description files, such as CompHEP's.

The current implementation of color flow amplitudes in PERL that makes multiple calls to O'Mega for computing the amplitude for each color flow will be replaced by a implementation of the same algorithm inside of O'Mega that will be much more efficient at compile time and will also yield a modest increase in execution speed for the resulting event generators.

8 Conclusions and Outlook

The LHC will take data from 2008 on, and physics and detector studies for the planned ILC are being refined with increasing requirements on the accuracy of theoretical predictions. Therefore, Monte Carlo simulation tools respond to the challenge to provide a flexibility and theoretical accuracy that will enable us to uncover the true nature of physics in the TeV energy range.

Multi-particle event generators such as WHIZARD will not completely replace well-established tools, but they are already indispensible for refining the accuracy of predictions, simulating complex elementary processes, and providing reliable background estimates where data alone are insufficient for unambiguous signal extraction.

Still missing are fully automated methods of incorporating higher orders of perturbation theory. While for inclusive quantities, analytic and semianalytic methods have proven successful, the situation for exclusive event generation is much worse, and it is likely that several approaches have to be combined and further developed before higher-order corrections can systematically be included in physics simulation.

WHIZARD covers elementary processes at all current and future high-energy colliders. The program is ready for use as a universal and flexible tool for experimental data analysis, data interpretation, and future phenomenological studies.

Acknowledgements

We would like to thank A. Alboteanu, T. Barklow, M. Beyer, E. Boos, R. Chierici, K. Desch, S. Dittmaier, T. Feldmann, T. Fritzsche, K. Hagiwara, T. Hahn, W. Hollik, M. Kobel, F. Krauss, P. Manakos, T. Mannel, M. Mertens, N. Meyer, K. Mönig, M. Moretti, D. Ondreka, M. Peskin, T. Plehn, D. Rainwater, H. Reuter, T. Robens, M. Ronan(†), S. Rosati, A. Rosca, J. Schumacher, M. Schumacher, S. Schumann, C. Schwinn, C. Speckner, and P. Zerwas for valuable discussions, comments and help during this project. WK and JR acknowledge the friendly atmosphere within and support by the particle physics groups at the University of Karlsruhe and DESY, Hamburg, and the Aspen Center for Physics, where a lot of this work has been initiated. JR wants especially to thank the particle physics group at Carleton University, Ottawa, where part of this work has been completed, for their warm hospitality and lots of intersting discussions. We would like to extend particular gratitude to C. Schwinn for his work on R_ξ-gauge functors and tests of gauge parameter independence in O'Mega amplitudes, and also for many helpful and enlightening discussions in an early stage of O'Mega.

This work has been supported in part by the Helmholtz-Gemeinschaft under Grant No. VH-NG-005, the Bundesministerium für Bildung und Forschung, Germany, (05 HT9RDA, 05

HA6VFB, 05 H4WWA/2) and the Deutsche Forschungsgemeinschaft (MA 676/6-1).

A Conventions for Polarized External States

In this appendix[12], we collect the conventions used in `WHIZARD` for the asymptotic states of polarized incoming and outgoing particles. The matrix elements generated by `O'Mega` call functions from `omegalib` that implement the following conventions. It is therefore starigh-forward to replace these conventions by another set, should the need ever arise in specialized applications.

A.1 On-shell wavefunctions

A.1.1 Dirac and Majorana fermions

We use the two-component Weyl spinors

$$\chi_+(\vec{p}) = \frac{1}{\sqrt{2|\vec{p}|(|\vec{p}|+p_3)}} \begin{pmatrix} |\vec{p}|+p_3 \\ p_1+\mathrm{i}p_2 \end{pmatrix} \qquad \chi_-(\vec{p}) \quad = \frac{1}{\sqrt{2|\vec{p}|(|\vec{p}|+p_3)}} \begin{pmatrix} -p_1+\mathrm{i}p_2 \\ |\vec{p}|+p_3 \end{pmatrix} \tag{5a}$$

to construct the four-component Dirac or Majorana spinors:

$$u_\pm(p) = \begin{pmatrix} \sqrt{p_0 \mp |\vec{p}|}\cdot\chi_\pm(\vec{p}) \\ \sqrt{p_0 \pm |\vec{p}|}\cdot\chi_\pm(\vec{p}) \end{pmatrix} \qquad v_\pm(p) = \begin{pmatrix} \mp\sqrt{p_0 \pm |\vec{p}|}\cdot\chi_\mp(\vec{p}) \\ \pm\sqrt{p_0 \mp |\vec{p}|}\cdot\chi_\mp(\vec{p}) \end{pmatrix} \tag{6}$$

For the implementation of purely Dirac fermions, there are also expressions for the conjugated spinors, which are not used in the mixed Dirac/Majorana implementation. There the conjugated spinors are constructed with the help of the charge-conjugation matrix.

A.1.2 Polarization vectors

We use the following conventions for spin-1 particles:

$$\epsilon_1^\mu(k) = \frac{1}{|\vec{k}|\sqrt{k_x^2+k_y^2}} \left(0; k_z k_x, k_y k_z, -k_x^2-k_y^2\right) \tag{7a}$$

$$\epsilon_2^\mu(k) = \frac{1}{\sqrt{k_x^2+k_y^2}} \left(0; -k_y, k_x, 0\right) \tag{7b}$$

$$\epsilon_3^\mu(k) = \frac{k_0}{m|\vec{k}|} \left(\vec{k}^2/k_0; k_x, k_y, k_z\right) \tag{7c}$$

[12] This appendix will be further extended in a future version to include more of the conventions used in `WHIZARD` and `O'Mega`; we hope that it will serve as a reference for making the usage and understanding of the program easier.

and

$$\epsilon^\mu_\pm(k) = \frac{1}{\sqrt{2}}(\epsilon^\mu_1(k) \pm \mathrm{i}\epsilon^\mu_2(k)) \tag{8a}$$

$$\epsilon^\mu_0(k) = \epsilon^\mu_3(k) \tag{8b}$$

i. e.

$$\epsilon^\mu_+(k) = \frac{1}{\sqrt{2}\sqrt{k_x^2+k_y^2}}\left(0; \frac{k_z k_x}{|\vec{k}|} - \mathrm{i}k_y, \frac{k_y k_z}{|\vec{k}|} + \mathrm{i}k_x, -\frac{k_x^2+k_y^2}{|\vec{k}|}\right) \tag{9a}$$

$$\epsilon^\mu_-(k) = \frac{1}{\sqrt{2}\sqrt{k_x^2+k_y^2}}\left(0; \frac{k_z k_x}{|\vec{k}|} + \mathrm{i}k_y, \frac{k_y k_z}{|\vec{k}|} - \mathrm{i}k_x, -\frac{k_x^2+k_y^2}{|\vec{k}|}\right) \tag{9b}$$

$$\epsilon^\mu_0(k) = \frac{k_0}{m|\vec{k}|}\left(\vec{k}^2/k_0; k_x, k_y, k_z\right) \tag{9c}$$

These conventions are similar to those used in `HELAS` [6], which are

$$\epsilon^\mu_\pm(k) = \frac{1}{\sqrt{2}}(\mp\epsilon^\mu_1(k) - \mathrm{i}\epsilon^\mu_2(k)) \tag{10a}$$

$$\epsilon^\mu_0(k) = \epsilon^\mu_3(k) \tag{10b}$$

with the same definitions as above.

Note that these conventions do not fit the definitions of the spinor wavefunctions defined in the last paragraph. In fact, they correspond to a different quantization axis for angular momentum. So, when constructing spin-3/2 wavefunctions out of those for spin 1/2 and spin 1, a different convention for the polarization vectors is used.

A.1.3 Polarization vectorspinors

The wavefunctions for (massive) gravitinos are constructed out of the wavefunctions of (massive) vectorbosons and (massive) Majorana fermions:

$$\psi^\mu_{(u;3/2)}(k) = \epsilon^\mu_+(k) \cdot u(k,+) \tag{11a}$$

$$\psi^\mu_{(u;1/2)}(k) = \sqrt{\frac{1}{3}}\,\epsilon^\mu_+(k) \cdot u(k,-) + \sqrt{\frac{2}{3}}\,\epsilon^\mu_0(k) \cdot u(k,+) \tag{11b}$$

$$\psi^\mu_{(u;-1/2)}(k) = \sqrt{\frac{2}{3}}\,\epsilon^\mu_0(k) \cdot u(k,-) + \sqrt{\frac{1}{3}}\,\epsilon^\mu_-(k) \cdot u(k,+) \tag{11c}$$

$$\psi^\mu_{(u;-3/2)}(k) = \epsilon^\mu_-(k) \cdot u(k,-) \tag{11d}$$

and in the same manner for $\psi^\mu_{(v;s)}$ with u replaced by v and with the conjugated polarization vectors. These gravitino wavefunctions obey the Dirac equation, they are transverse and they fulfill the irreducibility condition

$$\gamma_\mu \psi^\mu_{(u/v;s)} = 0. \tag{12}$$

As mentioned above, one needs to use the same quantization axis for spin 1/2 and spin 1 in order to construct the correct spin-3/2 states. The polarization vectors

$$\epsilon_+^\mu(k) = \frac{-e^{+\mathrm{i}\phi}}{\sqrt{2}}\,(0; \cos\theta\cos\phi - \mathrm{i}\sin\phi, \cos\theta\sin\phi + \mathrm{i}\cos\phi, -\sin\theta) \tag{13a}$$

$$\epsilon_-^\mu(k) = \frac{e^{-\mathrm{i}\phi}}{\sqrt{2}}\,(0; \cos\theta\cos\phi + \mathrm{i}\sin\phi, \cos\theta\sin\phi - \mathrm{i}\cos\phi, -\sin\theta) \tag{13b}$$

$$\epsilon_0^\mu(k) = \frac{1}{m}\left(|\vec{k}|; k^0\sin\theta\cos\phi, k^0\sin\theta\sin\phi, k^0\cos\theta\right) \tag{13c}$$

are used exclusively for this purpose.

A.1.4 Polarization tensors

Spin-2 polarization tensors are symmetric, transversal and traceless

$$\epsilon_m^{\mu\nu}(k) = \epsilon_m^{\nu\mu}(k) \tag{14a}$$

$$k_\mu\epsilon_m^{\mu\nu}(k) = k_\nu\epsilon_m^{\mu\nu}(k) = 0 \tag{14b}$$

$$\epsilon_{m,\mu}^{\mu}(k) = 0 \tag{14c}$$

with $m = 1, 2, 3, 4, 5$.

$$\epsilon_{+2}^{\mu\nu}(k) = \epsilon_+^\mu(k)\epsilon_+^\nu(k) \tag{15a}$$

$$\epsilon_{+1}^{\mu\nu}(k) = \frac{1}{\sqrt{2}}\left(\epsilon_+^\mu(k)\epsilon_0^\nu(k) + \epsilon_0^\mu(k)\epsilon_+^\nu(k)\right) \tag{15b}$$

$$\epsilon_0^{\mu\nu}(k) = \frac{1}{\sqrt{6}}\left(\epsilon_+^\mu(k)\epsilon_-^\nu(k) + \epsilon_-^\mu(k)\epsilon_+^\nu(k) - 2\epsilon_0^\mu(k)\epsilon_0^\nu(k)\right) \tag{15c}$$

$$\epsilon_{-1}^{\mu\nu}(k) = \frac{1}{\sqrt{2}}\left(\epsilon_-^\mu(k)\epsilon_0^\nu(k) + \epsilon_0^\mu(k)\epsilon_-^\nu(k)\right) \tag{15d}$$

$$\epsilon_{-2}^{\mu\nu}(k) = \epsilon_-^\mu(k)\epsilon_-^\nu(k) \tag{15e}$$

Here the polarization vectors from A.1.2 are used.

A.2 Propagators

Note that the sign of the momentum for fermionic lines is always negative because all momenta are treated as *outgoing* and the particle charge flow is therefore opposite to the momentum.

- Spin 0:

$$\frac{\mathrm{i}}{p^2 - m^2 + \mathrm{i}m\Gamma}\phi \tag{16}$$

- Spin 1/2:

$$\frac{i(-\not{p} + m)}{p^2 - m^2 + \mathrm{i}m\Gamma}\psi \qquad\qquad \bar{\psi}\frac{i(\not{p} + m)}{p^2 - m^2 + \mathrm{i}m\Gamma} \tag{17}$$

  The right one is only used for the pure Dirac implementation.

- Spin 1 (massive, unitarity gauge):

$$\frac{\mathrm{i}}{p^2 - m^2 + \mathrm{i}m\Gamma}\left(-g_{\mu\nu} + \frac{p_\mu p_\nu}{m^2}\right)\epsilon^\nu(p) \tag{18}$$

- Spin 1 (massless, Feynman gauge):

$$\frac{-i}{p^2}\epsilon^\nu(p) \tag{19}$$

- Spin 1 (massive, R_ξ gauge):

$$\frac{\mathrm{i}}{p^2}\left(-g_{\mu\nu} + (1-\xi)\frac{p_\mu p_\nu}{p^2}\right)\epsilon^\nu(p) \tag{20}$$

- Spin 3/2:

$$\frac{\mathrm{i}\left\{(-\not{p} + m)\left(-\eta_{\mu\nu} + \frac{p_\mu p_\nu}{m^2}\right) + \frac{1}{3}\left(\gamma_\mu - \frac{p_\mu}{m}\right)(\not{p} + m)\left(\gamma_\nu - \frac{p_\nu}{m}\right)\right\}}{p^2 - m^2 + \mathrm{i}m\Gamma}\,\psi^\nu \tag{21}$$

- Spin 2:

$$\frac{\mathrm{i}P^{\mu\nu,\rho\sigma}(p,m)}{p^2 - m^2 + \mathrm{i}m\Gamma}T_{\rho\sigma} \tag{22a}$$

with

$$P^{\mu\nu,\rho\sigma}(p,m) = \frac{1}{2}\left(g^{\mu\rho} - \frac{p^\mu p^\nu}{m^2}\right)\left(g^{\nu\sigma} - \frac{p^\nu p^\sigma}{m^2}\right) + \frac{1}{2}\left(g^{\mu\sigma} - \frac{p^\mu p^\sigma}{m^2}\right)\left(g^{\nu\rho} - \frac{p^\nu p^\rho}{m^2}\right) - \frac{1}{3}\left(g^{\mu\nu} - \frac{p^\mu p^\nu}{m^2}\right)\left(g^{\rho\sigma} - \frac{p^\rho p^\sigma}{m^2}\right) \tag{22b}$$

A.3 Vertices

For fermionic vertices we use the following chiral representation used in `HELAS` [6]:

$$\gamma^0 = \begin{pmatrix} 0 & \mathbf{1} \\ \mathbf{1} & 0 \end{pmatrix},\ \gamma^i = \begin{pmatrix} 0 & \sigma^i \\ -\sigma^i & 0 \end{pmatrix},\ \gamma_5 = i\gamma^0\gamma^1\gamma^2\gamma^3 = \begin{pmatrix} -\mathbf{1} & 0 \\ 0 & \mathbf{1} \end{pmatrix}, \tag{23a}$$

$$C = \begin{pmatrix} \epsilon & 0 \\ 0 & -\epsilon \end{pmatrix},\qquad \epsilon = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{23b}$$

The conventions for the fermionic vertices are shown in Table 3.

$\bar\psi(g_V\gamma^\mu - g_A\gamma^\mu\gamma_5)\psi$	`va_ff`$(g_V, g_A, \bar\psi, \psi)$	$\bar\psi(g_S + g_P\gamma_5)\psi$	`sp_ff`$(g_S, g_P, \bar\psi, \psi)$
$g_V\bar\psi\gamma^\mu\psi$	`v_ff`$(g_V, \bar\psi, \psi)$	$g_S\bar\psi\psi$	`s_ff`$(g_S, \bar\psi, \psi)$
$g_A\bar\psi\gamma_5\gamma^\mu\psi$	`a_ff`$(g_A, \bar\psi, \psi)$	$g_P\bar\psi\gamma_5\psi$	`p_ff`$(g_P, \bar\psi, \psi)$
$g_L\bar\psi\gamma^\mu(1-\gamma_5)\psi$	`vl_ff`$(g_L, \bar\psi, \psi)$	$g_L\bar\psi(1-\gamma_5)\psi$	`sl_ff`$(g_L, \bar\psi, \psi)$
$g_R\bar\psi\gamma^\mu(1+\gamma_5)\psi$	`vr_ff`$(g_R, \bar\psi, \psi)$	$g_R\bar\psi(1+\gamma_5)\psi$	`sr_ff`$(g_R, \bar\psi, \psi)$
$\not V(g_V - g_A\gamma_5)\psi$	`f_vaf`(g_V, g_A, V, ψ)	$\phi(g_S + g_P\gamma_5)\psi$	`f_spf`(g_S, g_P, ϕ, ψ)
$g_V\not V\psi$	`f_vf`(g_V, V, ψ)	$g_S\phi\psi$	`f_sf`(g_S, ϕ, ψ)
$g_A\gamma_5\not V\psi$	`f_af`(g_A, V, ψ)	$g_P\phi\gamma_5\psi$	`f_pf`(g_P, ϕ, ψ)
$g_L\not V(1-\gamma_5)\psi$	`f_vlf`(g_L, V, ψ)	$g_L\phi(1-\gamma_5)\psi$	`f_slf`(g_L, ϕ, ψ)
$g_R\not V(1+\gamma_5)\psi$	`f_vrf`(g_R, V, ψ)	$g_R\phi(1+\gamma_5)\psi$	`f_srf`(g_R, ϕ, ψ)
$\bar\psi\not V(g_V - g_A\gamma_5)$	`f_fva`$(g_V, g_A, \bar\psi, V)$	$\bar\psi\phi(g_S + g_P\gamma_5)$	`f_fsp`$(g_S, g_P, \bar\psi, \phi)$
$g_V\bar\psi\not V$	`f_fv`$(g_V, \bar\psi, V)$	$g_S\bar\psi\phi$	`f_fs`$(g_S, \bar\psi, \phi)$
$g_A\bar\psi\gamma_5\not V$	`f_fa`$(g_A, \bar\psi, V)$	$g_P\bar\psi\phi\gamma_5$	`f_fp`$(g_P, \bar\psi, \phi)$
$g_L\bar\psi\not V(1-\gamma_5)$	`f_fvl`$(g_L, \bar\psi, V)$	$g_L\bar\psi\phi(1-\gamma_5)$	`f_fsl`$(g_L, \bar\psi, \phi)$
$g_R\bar\psi\not V(1+\gamma_5)$	`f_fvr`$(g_R, \bar\psi, V)$	$g_R\bar\psi\phi(1+\gamma_5)$	`f_fsr`$(g_R, \bar\psi, \phi)$

Table 3: *Mnemonically abbreviated names of Fortran functions implementing fermionic vector and axial currents on the left, scalar and pseudoscalar currents on the right.*

References

[1] H. U. Bengtsson and T. Sjöstrand, Comput. Phys. Commun. **46**, 43 (1987); T. Sjöstrand, L. Lönnblad, S. Mrenna and P. Skands, arXiv:hep-ph/0308153.

[2] G. Marchesini, B. R. Webber, G. Abbiendi, I. G. Knowles, M. H. Seymour and L. Stanco, Comput. Phys. Commun. **67** (1992) 465; G. Corcella *et al.*, arXiv:hep-ph/0210213.

[3] A. Pukhov *et al.*, arXiv:hep-ph/9908288; E. Boos *et al.* [CompHEP Collaboration], Nucl. Instrum. Meth. A **534** (2004) 250 [arXiv:hep-ph/0403113].

[4] T. Ishikawa, T. Kaneko, K. Kato, S. Kawabata, Y. Shimizu and H. Tanaka [MINAMI-TATEYA group Collaboration], KEK-92-19; J. Fujimoto *et al.*, Comput. Phys. Commun. **153** (2003) 106 [arXiv:hep-ph/0208036].

[5] T. Stelzer, F. Long, Comput. Phys. Commun. **81** (1994) 357;

[6] H. Murayama, I. Watanabe and K. Hagiwara, KEK-91-11.

[7] R. Kleiss and R. Pittau, Comput. Phys. Commun. **83**, 141 (1994).

[8] T. Ohl, Comput. Phys. Commun. **120**, 13 (1999) [arXiv:hep-ph/9806432].

[9] F. Maltoni and T. Stelzer, JHEP **0302** (2003) 027;

[10] S. Frixione and B. R. Webber, JHEP **0206** (2002) 029 [arXiv:hep-ph/0204244]; S. Frixione, P. Nason and B. R. Webber, JHEP **0308** (2003) 007 [arXiv:hep-ph/0305252].

[11] S. Catani, F. Krauss, R. Kuhn and B. R. Webber, JHEP **0111** (2001) 063.

[12] F. Caravaglios and M. Moretti, Z. Phys. C **74**, 291 (1997) [arXiv:hep-ph/9604316].

[13] W. Kilian, LC-TOOL-2001-039; W. Kilian, *Prepared for 31st International Conference on High Energy Physics (ICHEP 2002), Amsterdam, The Netherlands, 24-31 Jul 2002.*

[14] A. Kanaki and C. G. Papadopoulos, Comput. Phys. Commun. **132**, 306 (2000) [arXiv:hep-ph/0002082].

[15] T. Gleisberg, S. Hoche, F. Krauss, A. Schalicke, S. Schumann and J. C. Winter, JHEP **0402**, 056 (2004).

[16] T. Ohl, arXiv:hep-ph/0011287; T. Ohl, arXiv:hep-ph/0011243; M. Moretti, T. Ohl and J. Reuter, arXiv:hep-ph/0102195.

[17] P. Skands *et al.*, JHEP **0407**, 036 (2004) [arXiv:hep-ph/0311123]; J. A. Aguilar-Saavedra *et al.*, Eur. Phys. J. C **46**, 43 (2006) [arXiv:hep-ph/0511344].

[18] Xavier Leroy, *The Objective Caml system, documentation and user's guide*, Technical Report, INRIA, 1997.

[19] M. Beneke, P. Falgari, C. Schwinn, A. Signer and G. Zanderighi, arXiv:0707.0773 [hep-ph]; C. Schwinn, arXiv:0708.0730 [hep-ph].

[20] S. Dittmaier and M. Roth, Nucl. Phys. B **642** (2002) 307 [arXiv:hep-ph/0206070].

[21] C. Schwinn, arXiv:hep-ph/0412028.

[22] A. Djouadi, W. Kilian, M. Muhlleitner and P. M. Zerwas, Eur. Phys. J. C **10** (1999) 27 [arXiv:hep-ph/9903229]; A. Djouadi, W. Kilian, M. Muhlleitner and P. M. Zerwas, Eur. Phys. J. C **10** (1999) 45 [arXiv:hep-ph/9904287].

[23] T. Barklow, talk at the American Linear Collider Physics Group Workshop, SLAC, Jan. 7–10, 2004.

[24] J. Hewett, talk at the International Linear Collider Workshop, DESY, May 30 – June 3, 2007.

[25] K. Cranmer, T. Plehn, J. Reuter, in preparation.

[26] J. Reuter, unpublished; M. Kuroda, KEK-CP-080, arXiv:hep-ph/9902340.

[27] A. Denner, H. Eck, O. Hahn and J. Küblbeck, Nucl. Phys. B **387**, 467 (1992); A. Denner, H. Eck, O. Hahn and J. Küblbeck, Phys. Lett. B **291**, 278 (1992).

[28] K. Hagiwara *et al.*, Phys. Rev. D **73**, 055005 (2006) [arXiv:hep-ph/0512260].

[29] J. Reuter, PhD thesis, TU Darmstadt 2002 , arXiv:hep-th/0212154.

[30] T. Ohl and J. Reuter, Eur. Phys. J. C **30**, 525 (2003) [arXiv:hep-th/0212224].

[31] C. Bartels, O. Kittel, U. Langenfeld, J. List, J. Reuter, in preparation.

[32] F. Deppisch, O. Kittel, J. Reuter, in preparation.

[33] J. Kalinowski, W. Kilian, J. Kovaric, J. Reuter, T. Robens, K. Rolbiecki, in preparation.

[34] A. Alboteanu, J. Alwall, D. Berdine, W. Kilian, T. Plehn, D. Rainwater, J. Reuter, S. Schumann, in preparation.

[35] W. Kilian and J. Reuter, Phys. Lett. B **642**, 81 (2006) [arXiv:hep-ph/0606277]; F. Deppisch, W. Kilian and J. Reuter, in preparation.

[36] W. Kilian and J. Reuter, Phys. Rev. D **70**, 015004 (2004) [arXiv:hep-ph/0311095]; W. Kilian, D. Rainwater and J. Reuter, Phys. Rev. D **71**, 015008 (2005) [arXiv:hep-ph/0411213]; *In the Proceedings of 2005 International Linear Collider Workshop (LCWS 2005), Stanford, California, 18-22 Mar 2005, pp 0109* [arXiv:hep-ph/0507081]; Phys. Rev. D **74**, 095003 (2006) [arXiv:hep-ph/0609119].

[37] J. Boersma, J. Reuter, in preparation.

[38] S. Heinemeyer *et al.*, arXiv:hep-ph/0511332; J. Reuter *et al.*, arXiv:hep-ph/0512012; B. C. Allanach *et al.*, arXiv:hep-ph/0602198; S. Kraml *et al.*, arXiv:hep-ph/0608079.

[39] E. Boos, H. J. He, W. Kilian, A. Pukhov, C. P. Yuan and P. M. Zerwas, Phys. Rev. D **57** (1998) 1553 [arXiv:hep-ph/9708310]; E. Boos, H. J. He, W. Kilian, A. Pukhov, C. P. Yuan and P. M. Zerwas, Phys. Rev. D **61** (2000) 077901 [arXiv:hep-ph/9908409].

[40] R. Chierici, S. Rosati and M. Kobel,

[41] W. Kilian and J. Reuter, *In the Proceedings of 2005 International Linear Collider Workshop (LCWS 2005), Stanford, California, 18-22 Mar 2005, pp 0311* [arXiv:hep-ph/0507099]; M. Beyer, W. Kilian, P. Krstonošic, K. Mönig, J. Reuter, E. Schmidt and H. Schröder, Eur. Phys. J. C **48**, 353 (2006) [arXiv:hep-ph/0604048].

[42] W. Kilian, M. Kobel, J. Reuter, J. Schumacher, in preparation.

[43] C. Schwinn, Phys. Rev. D **71** (2005) 113005 [arXiv:hep-ph/0504240].

[44] T. Ohl and J. Reuter, Phys. Rev. D **70**, 076007 (2004) [arXiv:hep-ph/0406098]; arXiv:hep-ph/0407337; A. Alboteanu, T. Ohl and R. Rückl, PoS **HEP2005** (2006) 322, [arXiv:hep-ph/0511188]; Phys. Rev. D **74** (2006) 096004, [arXiv:hep-ph/0608155].

[45] F. Cachazo, P. Svrcek and E. Witten, JHEP **0409**, 006 (2004) [arXiv:hep-th/0403047].

[46] M. Dinsdale, M. Ternick and S. Weinzierl, JHEP **0603**, 056 (2006) [arXiv:hep-ph/0602204].

[47] C. Schwinn and S. Weinzierl, JHEP **0603**, 030 (2006) [arXiv:hep-th/0602012].

[48] E. Boos and T. Ohl, Phys. Rev. Lett. **83**, 480 (1999) [arXiv:hep-ph/9903357]; E. Boos and T. Ohl, arXiv:hep-ph/9909487; T. Ohl and C. Schwinn, Eur. Phys. J. C **30**, 567 (2003) [arXiv:hep-ph/0305334].

[49] C. Schwinn, PhD thesis, TU Darmstadt/Univ. of Würzburg 2003, arXiv:hep-ph/0307057.

[50] F. Maltoni, K. Paul, T. Stelzer and S. Willenbrock, Phys. Rev. D **67** (2003) 014026 [arXiv:hep-ph/0209271].

[51] G. P. Lepage, CLNS-80/447.

[52] T. Ohl, Comput. Phys. Commun. **101**, 269 (1997); [arXiv:hep-ph/9607454].

[53] D. Schulte, CERN-PS-99-014-LP, CERN-PS-99-14-LP, CLIC-NOTE-387, CERN-CLIC-NOTE-387 (1999).

[54] H. Plothow-Besch, Comput. Phys. Commun. **75** (1993) 396.

[55] M. R. Whalley, D. Bourilkov and R. C. Group, arXiv:hep-ph/0508110; D. Bourilkov, arXiv:hep-ph/0305126.

[56] E. Boos *et al.*, arXiv:hep-ph/0109068.

[57] A. Semenov, Comput. Phys. Commun. **115**, 124 (1998).

[58] M. T. Ronan, in *Proc. of the APS/DPF/DPB Summer Study on the Future of Particle Physics (Snowmass 2001)* ed. N. Graf, *In the Proceedings of APS / DPF / DPB Summer Study on the Future of Particle Physics (Snowmass 2001), Snowmass, Colorado, 30 Jun - 21 Jul 2001, pp E3063.*

[59] W. Kilian, J. Reuter and T. Robens, Eur. Phys. J. C **48**, 389 (2006) [arXiv:hep-ph/0607127]; AIP Conf. Proc. **903**, 177 (2007) [arXiv:hep-ph/0610425].

[60] V. N. Gribov and L. N. Lipatov, Sov. J. Nucl. Phys. **15**, 438 (1972) [Yad. Fiz. **15**, 781 (1972)]; Sov. J. Nucl. Phys. **15**, 675 (1972) [Yad. Fiz. **15**, 1218 (1972)]; E. A. Kuraev and V. S. Fadin, Sov. J. Nucl. Phys. **41**, 466 (1985) [Yad. Fiz. **41**, 733 (1985)].

[61] M. Skrzypek and S. Jadach, Z. Phys. C **49**, 577 (1991).

[62] T. Fritzsche and W. Hollik, Nucl. Phys. Proc. Suppl. **135**, 102 (2004) [arXiv:hep-ph/0407095].

[63] T. Hahn, Comput. Phys. Commun. **140**, 418 (2001) [arXiv:hep-ph/0012260]. T. Hahn and C. Schappacher, Comput. Phys. Commun. **143**, 54 (2002) [arXiv:hep-ph/0105349]. T. Hahn and M. Perez-Victoria, Comput. Phys. Commun. **118**, 153 (1999) [arXiv:hep-ph/9807565].

[64] W. Kilian, J. Reuter, T Robens, in preparation.

[65] K. Mönig, J. Reuter, A. Rosca, in preparation.

[66] F. Krauss, JHEP **0208** (2002) 015; A. Schälicke and F. Krauss, JHEP **0507** (2005) 018.

[67] C. W. Bauer and M. D. Schwartz, Phys. Rev. Lett. **97** (2006) 142001 [arXiv:hep-ph/0604065]; C. W. Bauer and M. D. Schwartz, arXiv:hep-ph/0607296; C. W. Bauer and F. J. Tackmann, arXiv:0705.1719 [hep-ph].